Answering Multi-Dimensional Range Queries under Local Differential Privacy

Jianyu Yang[1,2*], Tianhao Wang[2], Ninghui Li[2], Xiang Cheng[1], Sen Su[1]
[1]*State Key Laboratory of Networking and Switching Technology,*
Beijing University of Posts and Telecommunications, Beijing, China
{jyyang, chengxiang, susen}@bupt.edu.cn
[2]*Department of Computer Science, Purdue University, West Lafayette, USA*
{yang1896, tianhaowang}@purdue.edu, ninghui@cs.purdue.edu

ABSTRACT

In this paper, we tackle the problem of answering multi-dimensional range queries under local differential privacy. There are three key technical challenges: capturing the correlations among attributes, avoiding the curse of dimensionality, and dealing with the large domains of attributes. None of the existing approaches satisfactorily deals with all three challenges. Overcoming these three challenges, we first propose an approach called Two-Dimensional Grids (TDG). Its main idea is to carefully use binning to partition the two-dimensional (2-D) domains of all attribute pairs into 2-D grids that can answer all 2-D range queries and then estimate the answer of a higher dimensional range query from the answers of the associated 2-D range queries. However, in order to reduce errors due to noises, coarse granularities are needed for each attribute in 2-D grids, losing fine-grained distribution information for individual attributes. To correct this deficiency, we further propose Hybrid-Dimensional Grids (HDG), which also introduces 1-D grids to capture finer-grained information on distribution of each individual attribute and combines information from 1-D and 2-D grids to answer range queries. To make HDG consistently effective, we provide a guideline for properly choosing granularities of grids based on an analysis of how different sources of errors are impacted by these choices. Extensive experiments conducted on real and synthetic datasets show that HDG can give a significant improvement over the existing approaches.

PVLDB Reference Format:
Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. Answering Multi-Dimensional Range Queries under Local Differential Privacy. PVLDB, 14(3): 378 - 390, 2021.
doi:10.14778/3430915.3430927

1 INTRODUCTION

Nowadays, users' data records contain many ordinal or numerical attributes in nature, e.g., income, age, the amount of time viewing a certain page, the number of times performing a certain actions, etc. The domains of these attributes consist of values that have a meaningful total order. A typical kind of fundamental analysis over users' records is multi-dimensional range query, which is a conjunction of multiple predicates for the attributes of interest and asks the fraction of users whose record satisfies all the predicates. In particular, a predicate is a restriction on the range of values of an attribute. However, users' records regarding these ordinal attributes are highly sensitive. Without strong privacy guarantee, answering multi-dimensional range queries over them will put individual privacy in jeopardy. Thus, developing effective approaches to address such privacy concerns becomes an urgent need.

In recent years, local differential privacy (LDP) has come to be the *de facto* standard for individual privacy protection. Under LDP, random noise is added on the client side before the data is sent to the central server. Thus, users do not need to rely on the trustworthiness of the central server. This desirable feature of LDP has led to wide deployment by industry (e.g., by Google [16], Apple [40], and Microsoft [11]). However, existing LDP solutions [9, 29, 46] are mostly limited to one-dimensional (1-D) range queries on a single attribute and cannot be well extended to handle multi-dimensional range queries.

In this paper, we tackle the problem of answering multi-dimensional range queries under LDP. Given a large number of users who have a record including multiple ordinal attributes, an untrusted aggregator aims at answering all possible multi-dimensional range queries over the users' records while satisfying LDP. To address the problem, we identify three key technical challenges: 1) how to capture the correlations among attributes, 2) how to avoid the curse of dimensionality, and 3) how to cope with the large domains of attributes. Any approach failing to solve any of these three challenges will have poor utility. As we show in Section 3, none of the existing approaches or their extensions can deal with all three challenges at the same time.

Overcoming these three challenges, we first propose an approach called Two-Dimensional Grids (TDG). Its main idea is to carefully use binning to partition the two-dimensional (2-D) domains of all attribute pairs into 2-D grids that can answer all possible 2-D range queries and then estimate the answer of a higher dimensional range query from the answers of the associated 2-D range queries. However, in order to reduce errors due to noises, coarse granularities are needed for each attribute in 2-D grids, losing fine-grained distribution information for individual attributes. When computing the answer of a 2-D range query by the cells that are partially included in the query, it needs to assume a uniform distribution within these cells, which may lead to large errors. To correct this deficiency, we further propose an upgraded approach called Hybrid-Dimensional

[*] Work done while studying as a visiting student at Purdue University.

Proceedings of the VLDB Endowment, Vol. 14, No. 3 ISSN 2150-8097.
doi:10.14778/3430915.3430927

Grids (HDG), whose core idea is combining hybrid dimensional (1-D and 2-D) information for better estimation. In particular, HDG also introduces 1-D grids to capture finer-grained information on distribution of each individual attribute and combines information from 1-D and 2-D grids to answer range queries. In both TDG and HDG, users are divided into groups, where each group reports information for one grid. After collecting frequencies of cells in each grid under LDP, the aggregator uses techniques to remove negativity and inconsistency among grids, and finally employs these grids to answer range queries.

However, it is still nontrivial to make HDG consistently effective, since the granularities for 1-D and 2-D grids can directly affect the performance of HDG. Consequently, it is essential to develop a method for determining the appropriate grid granularities so that HDG can guarantee the desirable utility. In particular in HDG, there are two main sources of errors: those due to noises generated by the random nature of LDP and those due to binning. When the distribution of values is fixed, errors due to binning do not change and can be viewed as bias because of the uniformity assumption, and errors due to noises can be viewed as variance. Thus choosing the granularities of grids can be viewed as a form of bias-variance tradeoff. Finer-grained grids lead to greater error due to noises, while coarser-grained ones result in greater error due to biases. The effect of each choice depends both on the privacy budget ε, population, and property of the distribution. By thoroughly analyzing the two sources of errors, we provide a guideline for properly choosing grid granularities under different parameter settings.

By capturing the necessary pair-wise attribute correlations via 2-D grids, both approaches overcome the first two challenges. Moreover, since they properly use binning with the provided guideline to reduce the error incurred by a large domain, the third challenge is carefully solved. Therefore, TDG usually performs better than the existing approaches. By also introducing 1-D grids to reduce the error due to the uniformity assumption, HDG can give a significant improvement over existing approaches.

Contributions. To summarize, this paper makes the following contributions:

- We propose TDG and HDG for answering multi-dimensional range queries under LDP, which include a guideline for choosing the grid granularities based on analysis of errors from different sources.
- We conduct extensive experiments to evaluate the performance of different approaches using both real and synthetic datasets. The results show that HDG outperforms existing approaches by one order of magnitude.

Roadmap. Section 2 provides the preliminaries. Section 3 describes the problem statement and four baseline approaches. Section 4 gives the details of our grid approaches. Section 5 shows our experimental results. Section 6 reviews related work. Finally, Section 7 concludes this paper.

2 PRELIMINARIES

2.1 Local Differential Privacy

Local differential privacy (LDP) [26] offers a high level of privacy protection, since each user only reports the sanitized data. Each user's privacy is still protected even if the aggregator is malicious. In particular, each user perturbs the value v using a randomized algorithm $\mathbf{A}$ and reports $\mathbf{A}(v)$ to the aggregator. Formally, LDP is defined in the following.

Definition 1 (Local Differential Privacy). *An algorithm* $\mathbf{A}(\cdot)$ *satisfies ε-local differential privacy (ε-LDP), where $\varepsilon \geq 0$, if and only if for any pair of inputs (v, v'), and any set* $\mathbf{R}$ *of possible outputs of* $\mathbf{A}$*, we have*

$$\Pr\left[\mathbf{A}(v) \in \mathbf{R}\right] \leq e^{\varepsilon} \Pr\left[\mathbf{A}(v') \in \mathbf{R}\right].$$

2.2 Categorical Frequency Oracles

In LDP, most problems can be reduced to frequency estimation. Below we present two state-of-the-art Categorical Frequency Oracle (CFO) protocols for these problems.

Randomized Response. The basic protocol in LDP is random response [49]. It was introduced for the binary case, but can be easily generalized to the categorical setting. Here we present the generalized version of random response (GRR), which enables the estimation of the frequency of any given value in a fixed domain.

Here each user with value $v \in [c]$ sends the true value v with probability p, and with probability $1-p$ sends a randomly chosen $v' \in [c]$ s.t. $v' \neq v$. More formally, the perturbation function is defined as

$$\forall_{y\in[c]} \Pr\left[\mathrm{GRR}(v) = y\right] = \begin{cases} p = \frac{e^{\varepsilon}}{e^{\varepsilon}+c-1}, & \text{if } y = v \\ p' = \frac{1}{e^{\varepsilon}+c-1}, & \text{if } y \neq v \end{cases} \tag{1}$$

This satisfies ϵ-LDP since $\frac{p}{p'} = e^{\varepsilon}$. To estimate the frequency of f_v for $v \in [c]$, one counts how many times v is reported, denoted by $\sum_{i\in[n]} \mathbb{1}_{\{y_i=v\}}$, and then computes $f_v = \frac{1}{n}\sum_{i\in[n]} \frac{\mathbb{1}_{\{y_i=v\}}-p'}{p-p'}$, where $\mathbb{1}_{\{y_i=v\}}$ is the indicator function that the report y_i of the i-th user equals v, and n is the total number of users.

Because each report y_i is an independent random variable, by the linearity of variance, we can show that the variance for this estimation is

$$\mathrm{Var}\left[f_v\right] = \frac{c-2+e^{\varepsilon}}{(e^{\varepsilon}-1)^2 \cdot n}. \tag{2}$$

Optimized Local Hash. The optimized local hash (OLH) protocol deals with a large domain by first using a hash function to compress the input domain $[c]$ into a smaller domain $[c']$, and then applying randomized response to the hashed value. In this protocol, both the hashing step and the randomization step result in information loss. The choice of the parameter c' is a trade-off between loss of information during the hashing step and loss of information during the randomization step. It is shown in [45] that the estimation variance as a function of c' is minimized when $c' = e^{\varepsilon} + 1$.

In OLH, one reports $\langle H, \mathrm{GRR}(H(v))\rangle$ where H is randomly chosen from a family of hash functions that hash each value in $[c]$ to a new one in $[c']$, and $\mathrm{GRR}(\cdot)$ is the perturbation function for random response, while operating on the domain $[c']$ (thus $p = \frac{e^{\varepsilon}}{e^{\varepsilon}+c'-1}$ in Equation (1)). Let $\langle H_i, y_i\rangle$ be the report from the i-th user. For each value $v \in [c]$, to compute its frequency, one first computes $|\{i \mid H_i(v) = y_i\}| = \sum_{i\in[n]} \mathbb{1}_{\{H_i(v)=y_i\}}$, and then transforms it to its unbiased estimation $f_v = \frac{1}{n}\sum\limits_{i\in[n]} \frac{\mathbb{1}_{\{H_i(v)=y_i\}}-1/c'}{p-1/c'}$.

In [45], it is shown that the estimation variance of OLH is

$$\mathsf{Var}\left[f_v\right] = \frac{4e^{\varepsilon}}{(e^{\varepsilon}-1)^2 \cdot n}. \tag{3}$$

Compared with GRR, OLH has a variance that does not depend on c. As a result, for a small c (such that $c-2 < 3e^{\varepsilon}$), GRR is better; but for a large c, OLH is preferable.

2.3 Principle of Dividing Users

Dividing users is one common feature among the existing LDP works [9, 46, 56]. That is, when multiple pieces of information are needed, the best results are obtained by dividing users into groups, and then gathering information from each group. This is different from the traditional DP setting [14], where there is a trusted aggregator having access to raw data records. In DP setting, the privacy budget is split to measure them all. This is because the estimation variance in LDP setting is linear in the number of users, while in DP setting, it is a constant. As a result, dividing users into m groups incurs a m^2 multiplicative factor in DP setting (because the result is multiplied by m), while in LDP setting, this factor is only m (because the number of users is divided by m). As splitting privacy budget by m increases variances for both cases by m^2, one prefers dividing users in LDP setting while splitting privacy budget in DP setting (as there is no sampling error). We will also apply this principle of dividing users in our proposed approaches.

3 PROBLEM STATEMENT AND BASELINE APPROACHES

3.1 Problem Statement

Consider there are d ordinal attributes $\{a_1, a_2, \cdots, a_d\}$. Without loss of generality, we assume that all attributes have the same domain $[c] = \{1, 2, \ldots, c\}$, where c is a power of two (if not in real setting, we can simply add some dummy values to achieve it). Let n be the total number of users. The i-th user's record is a d-dimensional vector, denoted by $\mathbf{v}_i = \langle v_i^1, v_i^2, \ldots, v_i^d \rangle$ where v_i^t means the value of attribute a_t in record $\mathbf{v}_i$.

We focus on the problem of answering multi-dimensional range queries under LDP. In particular, a multi-dimensional range query is a conjunction of multiple predicates for the attributes in its interest. Formally, a λ-dimensional (λ-D) range query q is defined as

$$q = (a_{t_1}, [l_{t_1}, r_{t_1}]) \wedge (a_{t_2}, [l_{t_2}, r_{t_2}]) \wedge \cdots \wedge (a_{t_\lambda}, [l_{t_\lambda}, r_{t_\lambda}]),$$

where $1 \le t_\phi \le d$, and $t_\phi \ne t_\psi$ when $\phi \ne \psi$. We define A_q to be $\{a_{t_\phi} | 1 \le \phi \le \lambda\}$ representing the set of attributes in q's interest. Intuitively, such a query q selects all records whose value of attribute a_{t_ϕ} is in the interval $[l_{t_\phi}, r_{t_\phi}]$ for all $a_{t_\phi} \in A_q$. The answer of the query q equals the fraction of these selected records. In particular, the real answer of q can be represented as

$$\bar{f}_q = \frac{|\{\mathbf{v}_i \mid v_i^t \in [l_t, r_t], \forall a_t \in A_q\}|}{n}.$$

In our problem setting, we assume that there is an aggregator that does not have access to the users' raw records. Our goal is to design an approach to enable the aggregator to get the answers of all possible range queries from the n users while satisfying LDP. Please see Table 1 for the list of notations.

Table 1: Notations

Notation	Meaning
n	The total number of users
d	The number of attributes
c	The domain size of an attribute
b	The branching factor of a hierarchy
m	The number of user groups
g	The granularity for an ordinal domain
q	The range query
A_q	The set of attributes in q's interest
λ	The query dimension

Key Technical Challenges. To address this problem, we identify three key technical challenges: 1) capturing the correlations among attributes, 2) avoiding the curse of dimensionality, and 3) coping with the large domains of attributes. Failure to solve any of these three challenges will lead to poor utility of the results.

In the following, we will describe four baseline approaches that may handle the problem of answering multi-dimensional range queries under LDP and analyze how they deal with these challenges. In particular, the first two approaches CALM and HIO are existing approaches that can be directly applied to this problem. The third approach Low-dimensional HIO (LHIO) is an improvement of HIO. The last approach Multiplied Square Wave (MSW) is an extension of the existing approach that may answer 1-D range queries.

3.2 CALM

CALM [56] is the state-of-the-art for marginal release under LDP. In particular, a λ-D marginal means the joint distribution of λ attributes. Due to the curse of dimensionality, directly computing a high-dimensional marginal using a LDP frequency oracle will lead to too much added noise. To solve this problem, CALM proposes to collect low-dimensional marginals and reconstruct a high-dimensional marginal from them. We notice that CALM can be used to answer range queries. In particular, for a λ-D range query, one can employ CALM to get its answer by directly summing up the noisy marginals included in the query.

CALM only captures necessary pair-wise attribute correlations, which effectively overcomes the first two challenges. However, it fails to solve the third challenge. To answer a range query, CALM needs to sum up all noisy marginals in the query, which may result in a large amount of noise in the answer when c is relatively large.

3.3 HIO

HIO [46] is a hierarchy-based approach that can directly answer multi-dimensional range queries under LDP. In HIO, given d attributes with the domain $[c]$, the aggregator first constructs a 1-D hierarchy for each attribute. To be specific, a 1-D hierarchy is a hierarchical collection of intervals with a branching factor b. The root corresponds to the entire domain $[c]$ and is recursively partitioned into b equally sized subintervals until the leaves whose corresponding subintervals only contain one value are reached.

Thus there are $h = \log_b c + 1$ levels, called one-dim levels, in a 1-D hierarchy. By defining that the root has a level 0, there are b^ℓ subintervals in a level $\ell \in [0, h]$. It is found in [46] that the optimal b is around 5. For illustration, we define a d-dim level as a group of d one-dim levels $(\ell_1, \ell_2, \ldots, \ell_d)$, each of which comes from one of these d 1-D hierarchies. Similarly, we define a d-dim interval as a group of d intervals, each of which also comes from one of these d 1-D hierarchies.

Then, the aggregator constructs a d-dimensional hierarchy with these d 1-D hierarchies. A level in the d-dimensional hierarchy is actually a d-dim level. Thus there are $(h+1)^d$ d-dim levels in the d-dimensional hierarchy. Since there are b^ℓ subintervals in a one-dim level ℓ in a 1-D hierarchy, a d-dim level $(\ell_1, \ell_2, \ldots, \ell_d)$ includes $\prod_{i=1}^{d} b^{\ell_i}$ d-dim intervals. Next, the aggregator randomly divides users into $(h+1)^d$ groups, where each group reports one d-dim level. After using OLH to get the noisy frequencies of all d-dim intervals in every d-dim level, the aggregator can answer a multi-dimensional range query in the following manner.

To answer a λ-D range query

$$q = (a_{t_1}, [l_{t_1}, r_{t_1}]) \wedge (a_{t_2}, [l_{t_2}, r_{t_2}]) \wedge \cdots \wedge (a_{t_\lambda}, [l_{t_\lambda}, r_{t_\lambda}]),$$

the aggregator first expands q to a new d-dimensional range query q' that is interested in all d attributes by assigning a specified interval $[1, c]$ for each attribute not in A_q. Then, for each attribute in these d attributes, the aggregator finds the least number of subintervals that can make up its specified interval in q' from its corresponding 1-D hierarchy. Finally, the aggregator sums up the noisy frequencies of all the d-intervals consisting of them to get the answer of q', which is equivalent to that of q.

HIO solves the first challenge by capturing the correlations among all attributes. However, HIO fails to handle the other two challenges. In HIO, users are divided into $(h+1)^d$ groups where $h = \log_b c$. When d or c is relatively large, there are too few users in each group, which will incur a high magnitude of added noise in the frequencies of d-dim intervals and result in large errors.

3.4 LHIO: Low-dimensional HIO

We observe that CALM achieves good utility by using 2-D marginals to reconstruct high-dimensional ones. Using this idea, we can modify HIO, resulting in a new approach called Low-dim HIO (LHIO). Its main idea is to compute the answers of 2-D range queries and then estimate the answer of a high-dimensional range query from them. Specifically, the aggregator first generates all $\binom{d}{2}$ attribute pairs from the given d attributes and then randomly divides users into $m = \binom{d}{2}$ groups, where each group works on one pair of attributes. Next, for each attribute pair, the aggregator invokes HIO to construct a 2-D hierarchy by interacting with its corresponding user group. The constructed 2-D hierarchies can be directly used to answer all possible 2-D range queries. To estimate the answer of a higher dimensional range query, the aggregator invokes the estimation method which will be presented in Section 4.4.

However, directly using the obtained noisy frequencies will lead to two inconsistency problems in our setting. The first one is within a 2-D hierarchy. That is, different levels of the noisy hierarchy may give inconsistent estimations due to LDP noise. The second one is among different 2-D hierarchies. Since each attribute is related to $d-1$ pairs, the frequencies marginalized on it from these $d-1$ 2-D hierarchy are usually different. The accuracy of the answers of the 2-D range queries will increase if the problem can be solved. We identify that the key to remove inconsistency is to solve the first inconsistency problem, since the second one can be easily solved by the overall consistency in CALM after the first one is handled. Therefore, we focus on the first problem and develop a new method to enforce consistency within a 2-D hierarchy. Its main idea is to adapt the constrained inference in Hay et al. [22] to a 2-D hierarchy and perform the operation twice by starting with the first and second attribute of the attribute pair, respectively. Its details are omitted due to space limitation.

LHIO satisfies ε-LDP because the report from each user uses OLH and satisfies ε-LDP. We show that by avoiding directly handling high-dimensional queries and removing inconsistency, LHIO can perform much better than HIO. Similar to CLAM, LHIO overcomes the first two challenges by capturing necessary pair-wise attribute correlations. However, LHIO fails to solve the third challenge. In LHIO, users are divided into $\binom{d}{2} \cdot (h+1)^2$ groups where $h = \log_b c$. For a relatively large c, it will also bring about excessive noises.

3.5 MSW: Multiplied Square Wave

Recently, Li et al. [29] proposed an approach called Square Wave (SW) for estimating the distribution of a single numerical attribute under LDP. It takes advantage of the ordinal nature of the domain and reports values that are close to the true value with higher probabilities than values that are farther away from the true value.

For handling an attribute with the discrete domain $[c]$, we initially normalize it to the continuous domain $[0, 1]$. Given a value $v \in [0, 1]$, SW perturbs it as:

$$\forall y \in [-\delta, 1+\delta],\ \Pr[\mathrm{SW}(v) = y] = \begin{cases} p, & \text{if } |v-y| \le \delta\,, \\ p', & \text{otherwise}\,, \end{cases}$$

where $\delta = \frac{\varepsilon e^{\varepsilon} - e^{\varepsilon} + 1}{2e^{\varepsilon}(e^{\varepsilon} - 1 - \varepsilon)}$ is the "closeness" threshold. By maximizing the difference between p and p' while satisfying that the total probability adds up to 1, the values p and p' can be derived as $p = \frac{e^{\varepsilon}}{2\delta e^{\varepsilon}+1}$ and $p' = \frac{1}{2\delta e^{\varepsilon}+1}$, respectively. After receiving perturbed reports from all users, the aggregator runs the Expectation Maximization algorithm to find an estimated distribution that maximizes the expectation of the observed output. It is shown in [29] that SW outperforms other approaches for answering 1-D range queries.

Here we introduce Multiplied Square Wave (MSW), which is extended from SW to handle multi-dimensional range queries under LDP. In MSW, given d attributes, the aggregator randomly divides users into d groups, where each group reports one attribute. After utilizing SW to obtain the distribution of each individual attribute, a multi-dimensional range query is answered by using the product of the answers of all associated 1-D range queries. Such approximation implicitly assumes that all attributes are independent.

In MSW, each user only reports one attribute via SW that satisfies ε-LDP. Therefore, this process can ensure ε-LDP for each user. In addition, the subsequent multiplication post-process steps take those outputs that are already differentially private and does not access any user's raw data. Thus, MSW satisfies ε-LDP. Since MSW only collects the information of individual attributes, it solves the last two challenge. However, it fails to handle the first challenge. MSW totally loses the correlations among attributes, which will produce high errors when handling correlated attributes.

4 GRID APPROACHES

In this section, we first elaborate our grid approaches for answering multi-dimensional range queries under LDP in Section 4.1-4.4. Then we give their privacy and utility analysis in Section 4.5. Finally, we describe how to choose the proper granularities in Section 4.6.

4.1 Overview

As analysed in Section 3, none of the baseline approaches can overcome all three key challenges. To address this problem, we first propose an approach called Two-Dimensional Grid (TDG). Its main idea is to carefully use binning to partition the 2-D domains of all attribute pairs into 2-D grids that can answer all 2-D range queries and then estimate the answer of a higher dimensional range query from the answers of the associated 2-D range queries.

However, since values within the same cell in a grid are reported together, the aggregator cannot tell the distribution within each cell and only assumes a uniform distribution. When computing the answer of a 2-D range query by the cells that are partially included in the query, this may lead to large error due to the uniformity assumption. To correct this deficiency, we further propose an upgraded approach called Hybrid-Dimensional Grid (HDG), which also introduces finer-grained 1-D grids and combines the information from 1-D and 2-D grids to answer range queries.

Note that the first two challenges pose a dilemma: capturing full correlations (as HIO) will lead to the curse of dimensionality; while only focusing on individual attributes (as MSW) will totally lose correlation information. In CALM [56], the similar dilemma is solved by using 2-D marginals to reconstruct high-dimensional ones, which achieves a good trade-off when handling these two challenges. Inspired by this idea, both TDG and HDG choose to capture the necessary pair-wise attribute correlations via 2-D grids, which overcomes the first two challenges. The third challenge is also carefully solved in TDG and HDG by properly using binning with the guideline to reduce the error incurred by a large domain.

Specifically, both TDG and HDG consist of three phases:

Phase 1. Constructing Grids. In TDG, from the given d attributes, the aggregator first generates all $\binom{d}{2}$ attribute pairs. Then the aggregator randomly divides users into $m = \binom{d}{2}$ groups, each of which corresponds to one pair. Next, for each attribute pair (a_j, a_k) where $1 \leq j < k \leq d$, the aggregator assigns the same granularity g_2 to construct a 2-D grid $G^{(j,k)}$ by partitioning the 2-D domain $[c] \times [c]$ into $g_2 \times g_2$ 2-D cells of equal size. In particular, each 2-D cell specifies a 2-D subdomain consisting of $\frac{c}{g_2} \times \frac{c}{g_2}$ 2-D values. Finally, to obtain noisy frequencies of cells in each grid, the aggregator instructs each user in the group corresponding to the grid to report which cell his/her private value is in using OLH.

In HDG, the aggregator also constructs d 1-D grids for the d attributes, respectively. Thus there will be $d + \binom{d}{2}$ grids in HDG and users are divided into $m = d + \binom{d}{2}$ groups, each of which corresponds to one of these grids. In addition to constructing $\binom{d}{2}$ 2-D grids with granularity g_2 as TDG, in HDG, the aggregator assigns the identical granularity g_1 to construct a 1-D grid $G^{(j)}$ containing g_1 1-D cells of equal size for each single attribute $a_j (1 \leq j \leq d)$. In particular, each 1-D cell specifies a 1-D subdomain consisting of $\frac{c}{g_1}$ 1-D values. Finally, as in TDG, the aggregator uses OLH to obtain noisy frequencies of cells in each grid.

Phase 2. Removing Negativity and Inconsistency. Due to using OLH to ensure privacy, the noisy frequency of a cell may be negative, which violates the prior knowledge that the true one is non-negative. Moreover, since an attribute is related to multiple grids, the noisy frequencies integrated on the attribute in different grids may be different, leading to inconsistency among grids. In this phase, to improve the utility, the aggregator post-processes the constructed grids to remove the negativity and inconsistency. The difference between TDG and HDG is that TDG only requires the aggregator to handle 2-D grids while 1-D and 2-D grids needs to be handled together in HDG. We describe the detail for post-processing grids in Section 4.2.

Phase 3. Answering Range Queries. In this phase, the aggregator can answer all multi-dimensional range queries. We first describe how to answer a 2-D range query. For ease of illustration, we take a 2-D range query q_0 interested in $A_{q_0} = \{a_1, a_2\}$ as an example. In TDG, to get the answer f_{q_0} of q_0, the aggregator first finds the 2-D grid $G^{(1,2)}$ corresponding to A_{q_0} and then checks all 2-D cells in $G^{(1,2)}$ in the following manner. If a cell is completely included in q_0, the aggregator includes its noisy frequency in f_{q_0}; if a cell is partially included, the aggregator estimates the sum of frequencies of common 2-D values between the cell and q_0 by uniform guess, i.e., assuming that the frequencies of 2-D values within the cell are uniformly distributed and then adds the sum to f_{q_0}.

In HDG, the aggregator treats those cells partially included in q_0 using a response matrix rather than uniform guess, which can significantly improve the accuracy of results. To be specific, for each attribute pair (a_j, a_k), the aggregator first employs the three grids $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$ to build a response matrix $M^{(j,k)}$ before answering 2-D range queries. In particular, the matrix $M^{(j,k)}$ consists of $c \times c$ elements that are in one-to-one correspondence with the estimated frequencies of 2-D values in the 2-D domain $[c] \times [c]$ of (a_j, a_k). The details of response matrix generation are given in Section 4.3. When calculating the answer f_{q_0} of the 2-D query q_0 in HDG, the aggregator also checks all 2-D cells in the grid $G^{(1,2)}$ corresponding to A_{q_0}. For a cell completely included in q_0, the aggregator includes its noisy frequency in f_{q_0} as in TDG; for a cell partially included in the query q_0, the aggregator identifies the common 2-D values between this cell and q_0, and then adds the sum of their corresponding elements in $M^{(1,2)}$ to f_{q_0}.

For a λ-D range query where $\lambda > 2$, its answer cannot be directly obtained from the constructed 2-D grids or response matrices. To answer this λ-D query, we propose to split it into $\binom{\lambda}{2}$ associated 2-D range queries and then estimate its answer from all answers of these $\binom{\lambda}{2}$ 2-D queries. We discuss it in detail in Section 4.4.

4.2 Post-Process for Grids

The post-process for grids contains two basic steps including non-negativity step and consistency step, which are used to remove negativity and inconsistency, respectively.

Non-Negativity Step. In this step, the aggregator handles the estimated frequencies of cells in each grid by Norm-Sub [48], which can make all estimates non-negative and sum up to 1. In Norm-Sub, firstly, all negative estimates are converted to 0. Then the total difference between 1 and the sum of positive estimates is calculated. Next, the average difference is obtained through dividing the total difference by the number of positive estimates. Finally, every positive estimate is updated by subtracting the average difference. The process is repeated until all estimates become non-negative.

Consistency Step. We first describe how to achieve consistency on an attribute among grids. For an attribute a, it is related to d grids in total, which includes one 1-D grid and $d-1$ 2-D grids. Assume these d grids are $\{G_1, G_2, \cdots G_d\}$. For an integer $j \in [1, g_2]$, we define $\mathrm{P}_{G_i}(a, j)$ to be the sum of frequencies of G_i's cells whose specified subdomain corresponds to a is in $[(j-1)\times\frac{c}{g_2}+1, j\times\frac{c}{g_2}]$. To make all $\mathrm{P}_{G_i}(a, j)$ consistent, we compute their weighted average as $\mathrm{P}(a, j) = \sum_{i=1}^{d} \theta_i \cdot \mathrm{P}_{G_i}(a, j)$, where θ_i is the weight of $\mathrm{P}_{G_i}(a, j)$.

To get a better estimation, we need to carefully set the value of θ_i. Our goal is to minimize the variance of $\mathrm{P}(a, j)$, i.e. $\mathrm{Var}\left[\mathrm{P}(a, j)\right] = \sum_{i=1}^{d} {\theta_i}^2 \cdot \mathrm{Var}\left[\mathrm{P}_{G_i}(a, j)\right] = \sum_{i=1}^{d} {\theta_i}^2 \cdot |S_i| \cdot \mathrm{Var}_0$, where S_i is the set of cells whose frequencies contribute to $\mathrm{P}_{G_i}(a, j)$ and Var_0 is the basic variance for estimating a single cell (we assume each user group has the same population). Apparently, if G_i is 1-D, $S_i = \frac{g_1}{g_2}$; if G_i is 2-D, $S_i = g_2$. Based on the analysis in [56], we have $\theta_i = \frac{1}{|S_i|} / \sum_{i=1}^{d} \frac{1}{|S_i|}$ and the optimal weighted average is $\mathrm{P}(a, j) = \left(\sum_{i=1}^{d} \frac{1}{|S_i|} \cdot \mathrm{P}_{G_i}(a, j)\right) / \sum_{i=1}^{d} \frac{1}{|S_i|}$. Once the $\mathrm{P}(a, j)$ is obtained, we need to make each $\mathrm{P}_{G_i}(a, j)$ equal it, which can be achieved in the following manner. For each cell in S_i, we update its frequency by adding the amount of change $\left(\mathrm{P}(a, j) - \mathrm{P}_{G_i}(a, j)\right) / |S_i|$.

To achieve consistency among all attributes, we can use the above method one by one for each single attribute. It is shown in [34] that following any order of these attributes, a later consistency step will not invalidate consistency established in previous steps.

Note that applying the consistency step may incur negativity, and vise versa. Thus in the post-process, we interchangeably invoke these two steps multiple times. Since we need to ensure non-negativity for the response matrix generation in Phase 3, we end the post-process with the non-negativity step. While the last step may again introduce inconsistency, it tends to be very small.

4.3 Response Matrix Generation

For an attribute pair (a_j, a_k), it corresponds to the response matrix $M^{(j,k)}$ of size $c \times c$, where the element $M^{(j,k)}[\beta_j, \beta_k]$ represents the estimated frequency of 2-D value (β_j, β_k) in the $[c] \times [c]$ 2-D domain of (a_j, a_k). To build $M^{(j,k)}$, we propose to invoke the efficient estimation method Weighted Update [2, 20] with the three grids $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$ corresponding to $\{a_j, a_k, (a_j, a_k)\}$, respectively. Its main idea is to keep using the information on each cell in these three grids to update the matrix until each cell's frequency equals the sum of its corresponding elements in the matrix.

Algorithm 1 Building Response Matrix

```
Input: Grids $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$, domain size $c$
Output: Response matrix $M^{(j,k)}$
 1: initialize all $c \times c$ elements in the matrix $M^{(j,k)}$ as $\frac{1}{c^2}$;
 2: repeat
 3:     for each grid $G$ in $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$ do
 4:         for each cell $s$ in $G$ do
 5:             Find the set of 2-D values $\Phi(s)$ corresponding to $s$;
 6:             Calculate $Y = \sum_{(\beta_j, \beta_k) \in \Phi(s)} M^{(j,k)}[\beta_j, \beta_k]$;
 7:             if $Y \neq 0$ then
 8:                 for each 2-D value $(\beta_j, \beta_k)$ in $\Phi(s)$ do
 9:                     $M^{(j,k)}[\beta_j, \beta_k] \leftarrow \frac{M^{(j,k)}[\beta_j, \beta_k]}{Y} \cdot f_s$;
10: until convergence
11: return $M^{(j,k)}$
```

Algorithm 1 provides the details of building response matrix $M^{(j,k)}$ for attribute pair (a_j, a_k). It takes grids $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$ and domain size c as inputs and outputs the response matrix $M^{(j,k)}$. In Algorithm 1, for each grid G in $\{G^{(j)}, G^{(k)}, G^{(j,k)}\}$, the aggregator performs the following update process on $M^{(j,k)}$. For each cell s in G, the aggregator first finds the set of 2-D values $\Phi(s)$ corresponding to s, which means that $\Phi(s)$ consists of all those 2-D values whose frequency can contribute to the frequency f_s of cell s. To illustrate the definition of $\Phi(s)$, we take a 2-D cell s in $G^{(j,k)}$ as an example. Assume the 2-D cell s specifies a 2-D subdomain $[l_s^j, r_s^j] \times [l_s^k, r_s^k]$, where $[l_s^j, r_s^j]$ and $[l_s^k, r_s^k]$ correspond to a_j and a_k, respectively. Then, $\Phi(s)$ can be represented as

$$\Phi(s) = \{(\beta_j, \beta_k) | \beta_j \in [l_s^j, r_s^j], \beta_k \in [l_s^k, r_s^k]\}.$$

Note that this representation is also applicable to a 1-D cell s in $G^{(j)}$ (or $G^{(k)}$), since we can equivalently transform its specified 1-D subdomain $[l_s^j, r_s^j]$ (or $[l_s^k, r_s^k]$) into 2-D subdomain $[l_s^j, r_s^j] \times [1, c]$ (or $[1, c] \times [l_s^k, r_s^k]$.). With $\Phi(s)$, the aggregator updates the elements in $M^{(j,k)}$ as Lines 6-9 in Algorithm 1. This update process is repeated until convergence.

In Algorithm 1, the convergence criteria is that the sum of the changes of all elements in the response matrix after each update process is lower than a given threshold. By comparing the results of setting different thresholds, we found that the results are almost the same so long as threshold is smaller than $\frac{1}{n}$.

4.4 Estimation for λ-D Range Query

To estimate the answer f_q of a λ-D range query

$$q = (a_{t_1}, [l_{t_1}, r_{t_1}]) \wedge (a_{t_2}, [l_{t_2}, r_{t_2}]) \wedge \cdots \wedge (a_{t_\lambda}, [l_{t_\lambda}, r_{t_\lambda}])$$

where $A_q = \{a_{t_\phi} | 1 \leq \phi \leq \lambda\}$, the aggregator first splits q into $\binom{\lambda}{2}$ associated 2-D range queries

$$\left\{q^{(j,k)} = (a_j, [l_j, r_j]) \wedge (a_k, [l_k, r_k]) | a_j, a_k \in A_q\right\},$$

and then gets their $\binom{\lambda}{2}$ answers $\left\{f_{q^{(j,k)}} \mid a_j, a_k \in A_q\right\}$ as described in Section 4.1. Finally, the aggregator uses these $\binom{\lambda}{2}$ 2-D queries' answers to estimate f_q.

In general, such an estimation problem can be solved by Maximum Entropy Optimization [34, 56]. Due to space limitation, we refer the readers to our full version [52] for its description. However, in experiments, we observe that Maximum Entropy Optimization cannot converge quickly in some cases. Therefore, we propose to use Weighted Update [2, 20] to solve this estimation problem, which can achieve almost the same accuracy while with higher efficiency.

Algorithm 2 gives the procedure of estimating the answer of a λ-D range query q. It takes the answers of $\binom{\lambda}{2}$ associated 2-D queries as inputs and outputs a estimated answer vector $\mathbf{z}$. In particular, the vector $\mathbf{z}$ consists of 2^λ elements that are in one-to-one correspondence with the answers of 2^λ λ-D queries in

$$Q(q) = \{\wedge_t(a_t, [l_t, r_t] \text{ or } [l_t, r_t]') \mid a_t \in A_q\},$$

where the interval $[l_t, r_t]'$ is the complement of $[l_t, r_t]$ on the domain of a_t. In Algorithm 2, for each $f_{q^{(j,k)}}$ in $\left\{f_{q^{(j,k)}} \mid a_j, a_k \in A_q\right\}$, the aggregator performs the following update process on $\mathbf{z}$. The aggregator first finds the set of λ-D queries $Q(q)^{(j,k)}$ corresponding to the 2-D query $q^{(j,k)}$, which means that $Q(q)^{(j,k)}$ consists of all those λ-D queries whose answer can contribute to $f_{q^{(j,k)}}$. In particular, $Q(q)^{(j,k)}$ contains $2^{\lambda-2}$ λ-D queries from $Q(q)$ and is defined as $\left\{\wedge_t(a_t, [l_t, r_t] \text{ or } [l_t, r_t]') \wedge q^{(j,k)} \mid a_t \in A_q/\{a_j, a_k\}\right\}$. Then, the aggregator calculates the sum Y of $\mathbf{z}[q']$ for all $q' \in Q(q)^{(j,k)}$, where $\mathbf{z}[q']$ is the element corresponding to the answer of q'. Next, the aggregator uses $f_{q^{(j,k)}}$ to update the elements in $\mathbf{z}$ as Lines 6-8. This process is repeated until convergence. The estimated answer f_q of the λ-D query q equals its corresponding element in $\mathbf{z}$, i.e., $\mathbf{z}[q]$.

In Algorithm 2, the convergence criteria is that the sum of the changes of all elements in the estimated vector after each update process is lower than a given threshold. We also found that the results are almost the same so long as threshold is smaller than $\frac{1}{n}$.

Algorithm 2 Estimating Answer of λ-D Range Query

```
Input: Associated 2-D queries' answers $\left\{f_{q^{(j,k)}} \mid a_j, a_k \in A_q\right\}$
Output: Estimated answer vector $\mathbf{z}$
 1: initialize all $2^\lambda$ elements in the vector $\mathbf{z}$ as $\frac{1}{2^\lambda}$;
 2: repeat
 3:     for each $f_{q^{(j,k)}}$ in $\left\{f_{q^{(j,k)}} \mid a_j, a_k \in A_q\right\}$ do
 4:         Find the set of queries $Q(q)^{(j,k)}$ corresponding to $q^{(j,k)}$;
 5:         Calculate $Y = \sum_{q' \in Q(q)^{(j,k)}} \mathbf{z}[q']$;
 6:         if $Y \neq 0$ then
 7:             for each query $q'$ in $Q(q)^{(j,k)}$ do
 8:                 $\mathbf{z}[q'] \leftarrow \frac{\mathbf{z}[q']}{Y} \cdot f_{q^{(j,k)}}$;
 9: until convergence
10: return z
```

4.5 Privacy and Utility Analysis

Privacy Guarantee. We claim that both TDG and HDG satisfy ε-LDP because all the information from each user to the aggregator goes through OLH with ε as privacy budget, and no other information is leaked.

Error Analysis. Below we analyze the expected squared error between the true query answer and the estimated answer. There are four kinds of errors: noise error, sampling error, non-uniformity error, and estimation error.

Noise and Sampling Error. The noise error is due to the use of LDP frequency oracles. To satisfy LDP, one adds, to each cell, an independently generated noise, and these noises have the same standard deviation. When summing up the noisy frequencies of cells to answer a query, the noise error is the sum of the corresponding noises. As these noises are independently generated zero-mean random variables, they cancel each other out to a certain degree. In fact, because these noises are independently generated, the variance of their sum equals the sum of their variances. Therefore, the finer granularity one partitions the domain into, the more cells are included in a query, and the larger the noise error is. The sampling error is incurred by using cells' frequencies obtained from a user group to represent those obtained from the entire population, since the user group may have different distribution from the global one.

The noise and sampling errors can be quantified together. Suppose the estimation is run on a sample D_η of the dataset D. We use $f_v(X)$ and $\bar{f}_v(X)$ to denote the estimated and true frequencies of v in X, respectively. For simplicity, the frequency on the original dataset $\bar{f}_v(D)$ is written as $\bar{f}_v$. The expected squared error for estimating one value is

$$\mathbf{E}\left[\left(f_v(D_\eta) - \bar{f}_v\right)^2\right] = \mathbf{E}\left[\left(f_v(D_\eta) - \bar{f}_v(D_\eta)\right)^2\right] + \mathbf{E}\left[\left(\bar{f}_v(D_\eta) - \bar{f}_v\right)^2\right] + 2\mathbf{E}\left[\left(f_v(D_\eta) - \bar{f}_v(D_\eta)\right) \cdot \left(\bar{f}_v(D_\eta) - \bar{f}_v\right)\right]. \quad (4)$$

Specifically, Equation (4) consists of three parts. The first part is the variance of frequency oracle, i.e.,

$$\mathbf{E}\left[\left(f_v(D_\eta) - \bar{f}_v(D_\eta)\right)^2\right] = m \cdot \frac{p'(1-p')}{n(p-p')^2} + m \cdot \frac{\bar{f}_v(p-p')(1-p-p')}{n(p-p')^2}.$$

In the case of OLH, we have $p = 1/2$, $p' = 1/(e^\varepsilon + 1)$, and the quantity equals $\frac{4me^\varepsilon}{n(e^\varepsilon - 1)^2} + \frac{m}{n} \cdot \bar{f}_v$.

The second part is $\mathbf{E}\left[\left(\bar{f}_v(D_\eta) - \bar{f}_v\right)^2\right] = \frac{m-1}{n-1}\bar{f}_v(1-\bar{f}_v)$. And the third part is $2\mathbf{E}\left[\left(f_v(D_\eta) - \bar{f}_v(D_\eta)\right) \cdot \left(\bar{f}_v(D_\eta) - \bar{f}_v\right)\right] = 0$. We observe that the second part is a constant which is much smaller than the first part. Ignoring the small factor $\frac{m}{n} \cdot \bar{f}_v$ in the first part, the expected squared noise and sampling error can be dominated by $\frac{4me^\varepsilon}{n(e^\varepsilon-1)^2}$. Due to space limitation, we refer the readers to our full version [52] for the detailed derivation of the above equations.

Non-Uniformity Error. Non-uniformity error is caused by cells that intersect with the query rectangle, but are not contained in it. For these cells, we need to estimate how many data points are in the intersected cells assuming that the data points are uniformly distributed, which will lead to non-uniformity error when the data points are not uniformly distributed. The magnitude of this error in any intersected cell, in general, depends on the number of data points in that cell, and is bounded by it. Therefore, the finer the partition granularity, the lower the non-uniformity error. Calculating precise non-uniformity error requires the availability of the true data distribution, which is not the case in our setting. Thus we opt to compute the approximate non-uniformity error.

Estimation Error. When estimating the answer of a λ-D range query where $\lambda > 2$ from the associated answers of 2-D range queries, estimation error will occur. Since the estimation error is dataset dependent, there is no formula for estimating it. In general, more accurate answers of 2-D range queries can result in a smaller estimation error. However, its feature that the magnitude is dependent on the dataset will introduce uncertainty, which means that an opposite result may appear in a few cases.

4.6 Choosing Granularities

Since the granularities g_1, g_2 can directly affect the utility of our gird approaches, we propose the following guideline for properly choosing them.

Guideline: To minimize the sum of squared noise and sampling error and squared non-uniformity error, the granularity g_1 for 1-D grids should be $g_1 = \sqrt[3]{\frac{n_1 \cdot (e^\varepsilon - 1)^2 \cdot \alpha_1{}^2}{2 m_1 e^\varepsilon}}$; the granularity g_2 for 2-D grids should be computed as $g_2 = \sqrt{2\alpha_2 \cdot (e^\varepsilon - 1) \cdot \sqrt{\frac{n_2}{m_2 e^\varepsilon}}}$, where ε is the total privacy budget, $n_i (i = 1, 2)$ is the number of users used for i-D grids, $m_i (i = 1, 2)$ is the number of user groups for i-D grids, and $\{\alpha_1, \alpha_2\}$ are some small constants depending on the dataset. For simplicity, we make each user group have the same population, i.e. $\frac{n_2}{m_2} = \frac{n}{\binom{d}{2}}$ for TDG and $\frac{n_1}{m_1} = \frac{n_2}{m_2} = \frac{n}{d + \binom{d}{2}}$ for HDG. To ensure that g_1 and g_2 are divisible by domain size c at the same time, for each of them, we take the power of two closest to its derived value as the final value. If the obtained granularity is larger than c, we set it to c by default. Our experimental results suggest that setting $\alpha_1 = 0.7$ and $\alpha_2 = 0.03$ can typically achieve good performance across different datasets.

Analysis on g_1. A range query on a 1-D grid specifies a query interval on the attribute corresponding to the grid. For an average case, we consider that the ratio of this interval to the attribute's domain size is $\frac{1}{2}$. When answering the query from a 1-D grid with granularity g_1, there are roughly $\frac{g_1}{2}$ cells included in this query. The squared noise and sampling error is $\frac{g_1}{2} \cdot \frac{4 m_1 e^\varepsilon}{n_1 (e^\varepsilon - 1)^2} = \frac{2 g_1 m_1 e^\varepsilon}{n_1 (e^\varepsilon - 1)^2}$.

The non-uniformity error is proportional to the sum of frequencies of values in the cells that intersect with the two sides of the query interval. Assuming that the non-uniformity error is $\frac{\alpha_1}{g_1}$ for some constant α_1, then it has a squared error of $\left(\frac{\alpha_1}{g_1}\right)^2$.

The minimize the sum of the two squared errors $\frac{2 g_1 m_1 e^\varepsilon}{n_1 (e^\varepsilon - 1)^2} + \left(\frac{\alpha_1}{g_1}\right)^2$, we should set g_1 to $\sqrt[3]{\frac{n_1 \cdot (e^\varepsilon - 1)^2 \cdot \alpha_1{}^2}{2 m_1 e^\varepsilon}}$.

Analysis on g_2. Here we extend the above analysis to the 2-D grid setting. For a 2-D query, we assume that the ratio of each query interval to its corresponding attribute's domain size is $\frac{1}{2}$. Then the squared noise and sampling error is $(\frac{g_2}{2})^2 \cdot \frac{4 m_2 e^\varepsilon}{n_2 (e^\varepsilon - 1)^2} = \frac{(g_2)^2 \cdot m_2 e^\varepsilon}{n_2 (e^\varepsilon - 1)^2}$.

The non-uniformity error is proportional to the sum of the frequencies of values in the cells that fall on the four edges of the query rectangle. The query rectangle's edges contain $4 \cdot \frac{g_2}{2} = 2 g_2$ cells; and the expected sum of frequencies of values included in these cells is $2 g_2 \cdot \frac{1}{g_2 \times g_2} = \frac{2}{g_2}$. Similar to the 1-D grid setting, we assume that the non-uniformity error on average is some portion of it. Then the squared error from non-uniformity is $\left(\frac{2\alpha_2}{g_2}\right)^2$ for some constant α_2. Our goal is to select g_2 to minimize the sum of the two squared errors $\frac{(g_2)^2 \cdot m_2 e^\varepsilon}{n_2 (e^\varepsilon - 1)^2} + \left(\frac{2\alpha_2}{g_2}\right)^2$. To achieve this goal, g_2 should be $\sqrt{2\alpha_2 \cdot (e^\varepsilon - 1) \cdot \sqrt{\frac{n_2}{m_2 e^\varepsilon}}}$.

Discussion. In the analysis of non-uniformity error, for a cell that contributes to this error, we calculate the expected sum of frequencies of values in this cell based on the uniformity assumption. Although this assumption may lead to the deviation between the calculated error and the true one, it helps the analysis become more general for diverse datasets. Moreover, since 1-D grids are finer-grained, this deviation's influence on the performance of HDG tends to be negligible. Thus, such an assumption still makes our guideline consistently effective for HDG when handling diverse datasets. Note that the recommended values of $\{\alpha_1, \alpha_2\}$ are obtained by tuning them on synthetic datasets under different setting of n, c, d, which does not leak any real users' private information. Besides, all other necessary parameters for choosing granularities are derived from public background knowledge and do not require the aggregator to access raw data. Therefore, configuring TDG and HDG with our guideline will not lead to any privacy leakage.

5 EXPERIMENTAL EVALUATION

In this section, we aim to answer the following questions: (1) how does our proposed HDG perform, (2) how can different parameters affect the results, and 3) how effective is the guidance for choosing granularities given by our guideline.

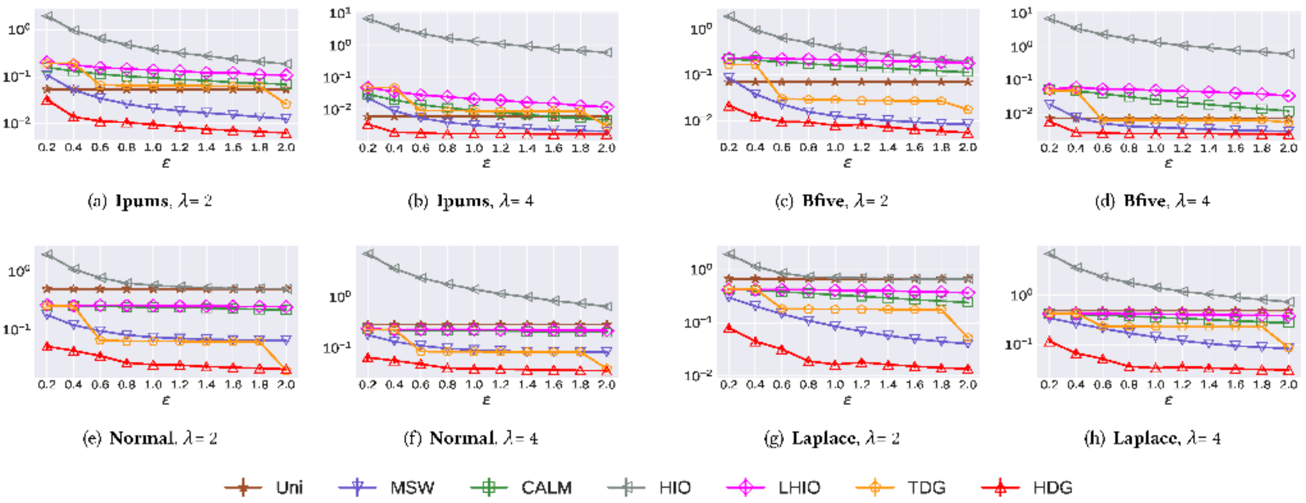
5.1 Setup

Datasets. We make use of two real datasets and two synthetic datasets in our experiments.

- Ipums [38]: It is from the Integrated Public Use Microdata Series and has around 1 million records of the United States census in 2018.
- Bfive [25]: It is collected through an interactive on-line personality test and contains around 1 million records. Each record describes the time spent on each question in milliseconds.
- Normal: This dataset is synthesized from multivariate normal distribution with mean 0, standard deviation 1. The covariance between every two attributes is 0.8.
- Laplace: This dataset is synthesized from multivariate laplace distribution with mean 0, standard deviation 1. The covariance between every two attributes is 0.8.

For the first two real datasets, we sample 1 million user records. To experiment with different numbers of users, we generate multiple test datasets from the two synthetic datasets with the number of users ranging from 100k to 10M. For evaluation varying different numbers of attributes and domain sizes, we generate multiple versions of these four datasets with the number of attributes ranging from 3 to 10 and their domain sizes ranging from 2^4 to 2^{10}.

Competitors. We compare HDG against TDG and all the baseline approaches including HIO, CALM, MSW and LHIO. In addition, we add a benchmark approach Uni which always outputs a uniform guess. In particular, we set the branch factor $b = 4$ for HIO and LHIO.



(a) **Ipums**, λ= 2

(b) **Ipums**, λ= 4

(c) **Bfive**, λ= 2

(d) **Bfive**, λ= 4

(e) **Normal**, λ= 2

(f) **Normal**, λ= 4

(g) **Laplace**, λ= 2

(h) **Laplace**, λ= 4

Figure 1: Varying ε on all datasets under setting of n= 10^6, d= 6, c= 64, ω= 0.5, λ= 2, 4. MAEs are shown in log scale.

For CALM, we choose to reconstruct high-dimensional marginals from 2-D ones. To configure TDG and HDG with our guideline, we first set the recommended $\alpha_1 = 0.7$ and $\alpha_2 = 0.03$. Then, for handling a dataset, we use its public information including the number of users n and the number of attributes d to obtain the (n_i, m_i) $(i = 1, 2)$ according to the provided strategy in our guideline. Finally, given a privacy budget ε, we derive the values of g_1 and g_2 from the equations in our guideline. Note that except for HIO and Uni, all other approaches contain the consistency operation inside.

Utility Metric. We use the Mean Absolute Error (MAE) to measure the accuracy of estimated answers. Given a set Q of range queries, it is computed as MAE = $\frac{1}{|Q|}\sum_{q\in Q}|f_q - \tilde{f}_q|$, where f_q and $\tilde{f}_q$ are the estimated and true answers of query q, respectively.

Methodology. To evaluate the performance of HDG, we randomly select a set Q of λ-D range queries and calculate their MAE. We generate range queries with different dimensional query volumes denoted by ω, which means the ratio of the specified interval to the domain size for each queried attribute. In all subsequent experiments, unless explicitly stated, we use the following default values for other relevant parameters: $\varepsilon = 1.0$, $\omega = 0.5$, $d = 6$, $c = 64$, $n = 10^6$, $\lambda = \{2, 4\}$ and $|Q| = 200$.

We implemented all approaches using Python3.7. The source code of our approaches is publicly available at [53]. All experiments were conducted on servers running Linux kernel version 5.0 with Intel Xeon Silver 4108 CPU @ 1.80GHz and 128GB memory. For each dataset and each approach, we repeat each experiment 10 times and report result mean and standard deviation. Note that standard deviation is invisible in most cases because the performance is stable in our results. Besides, due to high MAEs, the results of HIO are automatically omitted in some figures for more noticeable differences among other approaches.

5.2 Overall Results

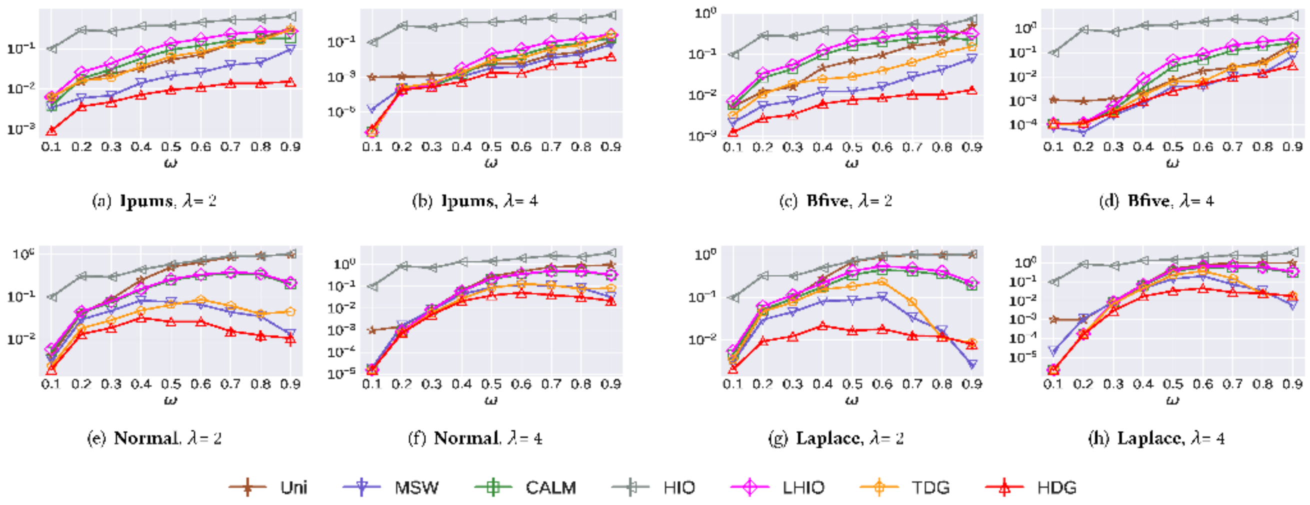
Figure 1 shows the results for comparing HDG against all the competitors under different ε on all four datasets. As expected, we can observe that except Uni, the accuracy of all other approaches becomes better (value of MAE gets lower) when ε grows. Among these approaches, HIO performs the worst, even worse than Uni in most of the cases. On all datasets, our improved LHIO performs roughly one order of magnitude better than HIO in the low ε region, but the improvement is less significant for a larger ε. This is because when ε is small, the consistency step of LHIO corrects many inconsistency; and when ε gets larger, the error becomes less and so is the effect of the consistency step. Moreover, CALM performs better than LHIO. The reason is that LHIO have much fewer users in each group than CALM under this setting, which incurs excessive noise canceling out the benefit of hierarchy. We also observe that MSW can achieve a high accuracy on Bfive dataset (Figure 1(c) and (d)), which indicates that the correlations among the attributes in Bfive dataset are weak. But we can see that the utility of HDG is still comparable to MSW, which confirms that HDG can also handle the datasets with low correlation well.

Figure 1 shows that TDG and HDG have a clear advantage over other approaches; and HDG performs better than TDG. Note that there are some jumping points of the two approaches. This is because HDG and TDG choose different granularities based on ε values and dataset sizes, and the choices, while generally good, are not optimal for every dataset at every ε value.

5.3 Impact of Different Parameters

In this part, we compare different approaches under different parameter settings. In general, these parameters including the dimensional query volume ω, the domain size c of an attribute, the number of attributes d, the query dimension λ and the total number of users n can also affect the performance of the approaches.

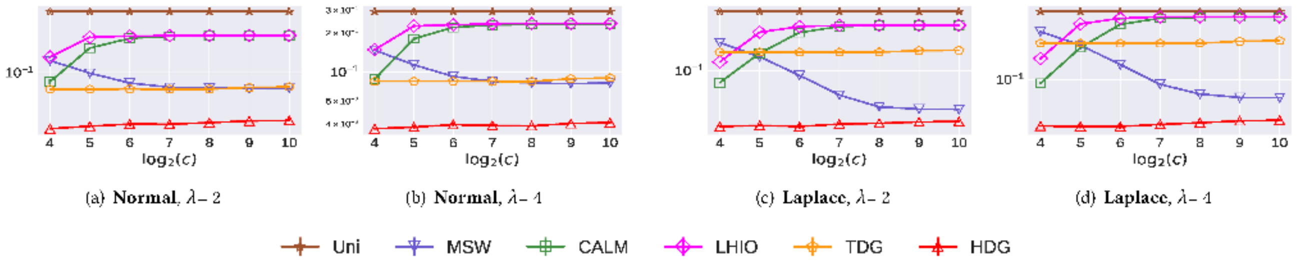
The impact of ω. Figure 2 shows the results varying ω from 0.1 to 0.9. From Figure 2, we can observe that HDG can consistently outperform all other approaches. In general, for all approaches, their utilities degrade when ω increases. It is because there are more cells included in the range query and the noise error incurred by enforcing LDP grows. Moreover, we can observe that except for



(a) **Ipums**, λ= 2 (b) **Ipums**, λ= 4 (c) **Bfive**, λ= 2 (d) **Bfive**, λ= 4

(e) **Normal**, λ= 2 (f) **Normal**, λ= 4 (g) **Laplace**, λ= 2 (h) **Laplace**, λ= 4

Figure 2: Varying ω on all datasets under setting of n= 10^6, d= 6, c= 64, ϵ= 1.0, λ= 2, 4. MAEs are shown in log scale.



(a) **Normal**, λ= 2 (b) **Normal**, λ= 4 (c) **Laplace**, λ= 2 (d) **Laplace**, λ= 4

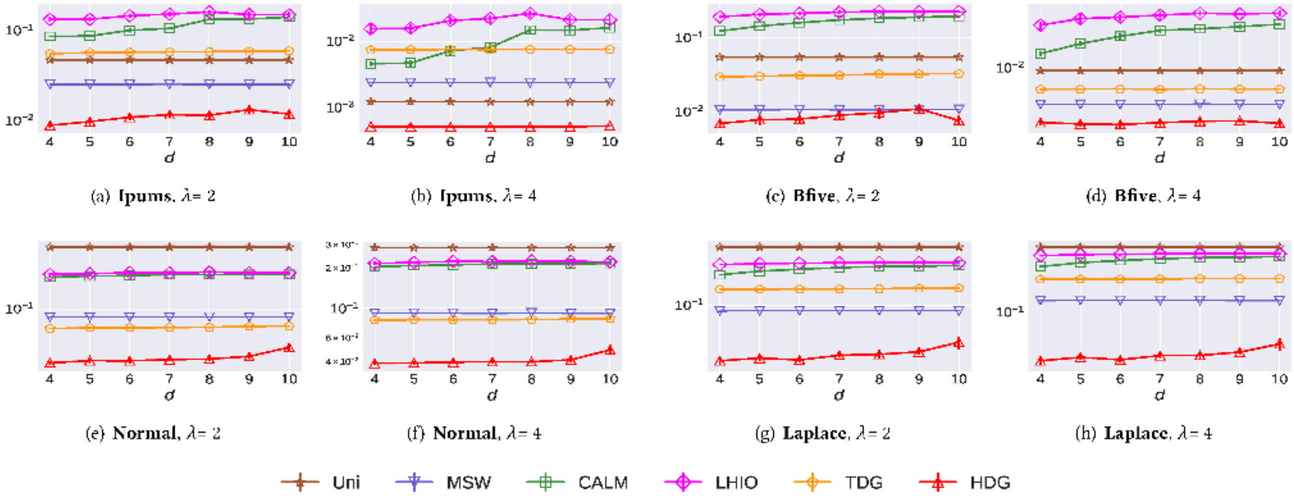
Figure 3: Varying c on synthetic datasets under setting of n= 10^6, d= 6, ϵ= 1.0, ω= 0.5, λ= 2, 4. MAEs are shown in log scale.

HIO, all LDP approaches have arch-like MAE trends, which means that their MAEs first increase and then decrease as ω increases. This is due to the consistency operation, also observed in [48]. In particular, when the queried area gets larger, with the enforcement of the consistency that the frequencies sum up to 1, the result is essentially 1 minus the un-queried areas.

The impact of c. Figure 3 presents the results varying c from 2^4 to 2^{10} on synthetic datasets. We can observe that HDG performs the best among all approaches. Moreover, the utility of HDG remains stable when c becomes larger. It is because that the noise error and non-uniformity error do not change a lot for a grid as c changes. As expected, the MAEs of CALM and LHIO become higher as c increases, which is consistent with our analysis that more marginals included in the query lead to more LDP noise in the answer. We also find that MSW achieves higher utility when c grows. That is because its advantage of reporting values that are close to the true value with higher probabilities becomes more pronounced, especially for the Laplace dataset with spike distribution.

The impact of d. Figure 4 gives the results varying d from 3 to 10. The relative order of different approaches are the same as we have already observed previously. We can observe that the MAEs of an LDP approach basically become higher when d increases. The reason is that for a larger d, there are more user groups and fewer users in each group, which makes the amount of noise and sampling errors grow. In addition, we find an outlier at d = 10 in Figure 4(c) where the HDG's MAE at d = 10 are smaller than those at d = 9. This is due to the changes of the granularities. In particular, when d increases from 9 to 10, the suggested granularities change from (16, 4) to (16, 2), which are more appropriate for Bfive dataset.

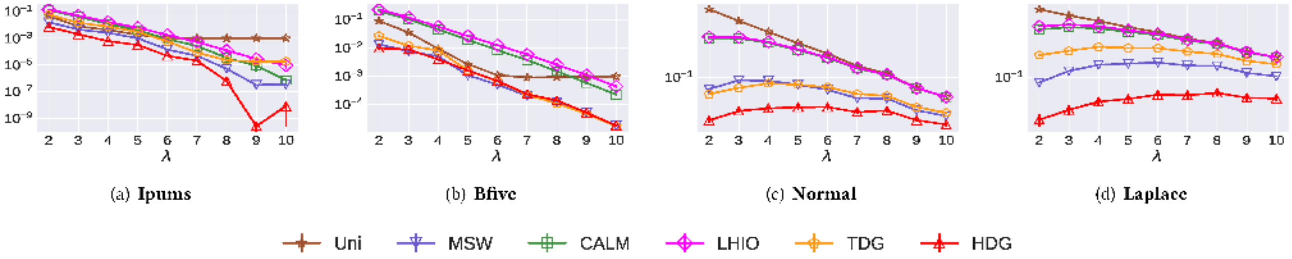
The impact of λ. Figure 5 studies the impact of λ on the utility of each approach. We observe that the MAEs of LDP approaches decrease as λ increases on real datasets (Figure 5(a) and (b)). On synthetic datasets, the MAEs first grow and then drop along with the increment of λ (Figure 5(c) and (d)). The reason can be explained as follows. Intuitively, when λ becomes larger, there will be more estimation error included in the estimated answers. It is why the MAEs gradually grow at the beginning on synthetic datasets. However, for a relatively large λ, the true answer of a λ-D range query is close to zero. Due to the large amount of estimation error, the post-progress for removing negativity and inconsistency can also make the estimated answers approach zero and thus the MAEs are reduced. On real datasets, the effect of post-progress plays a decisive role since λ = 3. We also find an outlier at λ = 10 in Figure 5(a) where the HDG's MAE at λ = 10 are higher than that at λ = 9. The reason is that estimation error occurs when answering high dimensional range queries, and its feature that the magnitude is dependent on the dataset introduces uncertainty as mentioned in Section 4.5.



(a) **Ipums**, λ= 2 (b) **Ipums**, λ= 4 (c) **Bfive**, λ= 2 (d) **Bfive**, λ= 4

(e) **Normal**, λ= 2 (f) **Normal**, λ= 4 (g) **Laplace**, λ= 2 (h) **Laplace**, λ= 4

Figure 4: Varying d on all datasets under setting of n= 10^6, c= 64, ε= 1.0, ω= 0.5, λ= 2, 4. MAEs are shown in log scale.



(a) **Ipums** (b) **Bfive** (c) **Normal** (d) **Laplace**

Figure 5: Varying λ on all datasets under setting of n= 10^6, d= 6, c= 64, ε= 1.0, ω= 0.5. MAEs are shown in log scale.

The impact of n. Figure 6 shows the results varying n from 100K to 10M on synthetic datasets. Not surprisingly, for the approaches satisfying LDP, larger population can boost the accuracy of their results. We can observe that HDG consistently achieves the best performance among all approaches. It can be expected that when applying HDG to real-world applications where the population is large, we are able to achieve desirable performance.

5.4 Effectiveness of Guideline

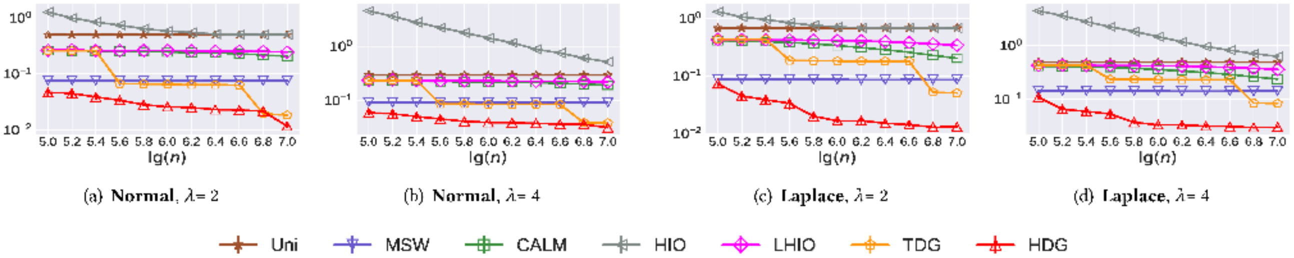
To evaluate the effectiveness of our proposed guideline for choosing granularities in HDG, we first enumerate all possible combinations of g_1 and g_2 for a given domain size c. Then, for each combination (g_1, g_2), we use it as the chosen granularities to implement a version of HDG, which is referred to as HDG(g_1, g_2). The approach labeled by HDG adopts granularities obtained from our proposed guideline under the suggested setting $\alpha_1 = 0.7$ and $\alpha_2 = 0.03$. Finally, we compare HDG with all the implemented versions to judge whether our guideline can provide good choices of granularities under different settings.

Figure 7 shows the results on 2-D range queries, which can avoid the the influence of estimation error. From Figure 7, we can see throughout the four datasets, HDG performs reasonably well for all ε values. Although HDG may not perform best all the time, it can consistently achieve a very close accuracy to the best performing version, which confirms that our guideline can always give helpful guidance. We have also evaluated the effectiveness of our guideline under different n, c and d; the results give similar conclusion, and are omitted due to space limitation.

We have also conducted several sets of experiments to confirm the effectiveness of Phase 2 in HDG, to validate the effectiveness of the recommended parameter settings in our guideline, to verify the efficient convergence of Algorithms 1 and 2, and to demonstrate that the superiority of HDG also holds for other datasets. Due to space limitation, we refer the readers to our full version [52] for these experimental results and their analysis.

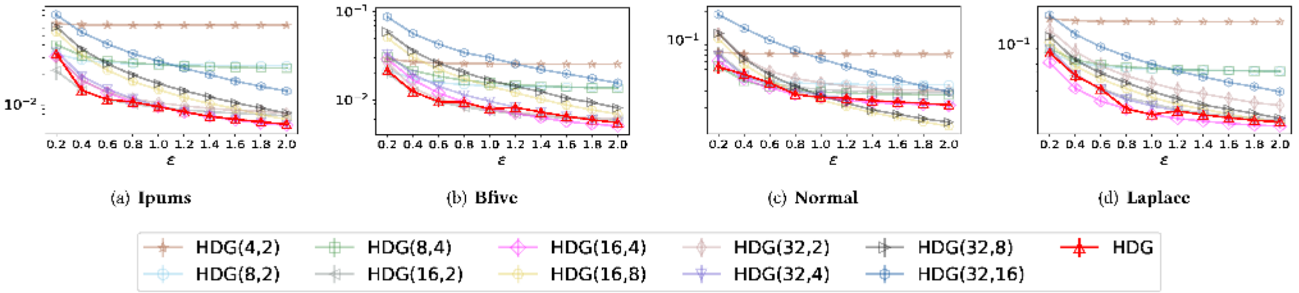
6 RELATED WORK

Range queries have been widely studied under traditional DP [14]. Xiao et al. [50] propose a framework Privelet, which employs wavelet transforms such as Haar wavelet to handle range queries. Hay et al. [22] introduce the hierarchical intervals technique accompanied by constrained inference for ensuring consistency. Cormode et al. [10] utilize indexing methods such as quadtrees and kd-trees to generate spatial decompositions for describing the data distribution. Qardaji et al. [33] provide a better understanding of using hierarchical methods for histogram publication. Li et al. [27] propose a two-stage approach DAWA utilizing a variant of the exponential mechanism to partition the domain into uniform regions in the first stage, which cannot be done in LDP setting. Qardaji et al. [32]



(a) **Normal**, λ= 2 (b) **Normal**, λ= 4 (c) **Laplace**, λ= 2 (d) **Laplace**, λ= 4

Figure 6: Varying n on synthetic datasets under setting of d= 6, c= 64, ε= 1.0, ω= 0.5, λ= 2, 4. MAEs are shown in log scale.



(a) **Ipums** (b) **Bfive** (c) **Normal** (d) **Laplace**

Figure 7: Verifying guideline in HDG under setting of n= 10^6, d= 6, c= 64, ω= 0.5, λ= 2. MAEs are shown in log scale.

present an Adaptive Grids (AG) approach to release a synopsis for 2-D geospatial data and show that AG can perform better than those hierarchy approaches. Note that the idea of grid is also adopted in our approach HDG, but there are several differences between HDG and AG. First, AG is only for 2-D data, and HDG is for multi-dimensional data, combining information from many 2-D grids. Moreover, for the first time, HDG proposes to combine information on both 1-D and 2-D grids to answer range queries. Finally, due to the feature of LDP setting, our HDG collects the information on grids by dividing users rather than the privacy budget and thus gives a novel analysis of different sources of errors and guideline. For standardized evaluation of differential private algorithms that answer 1-D and 2-D range queries, Hay et al. [21] propose a novel evaluation framework DPBench. McKenna et al. [31] describe an algorithm HDMM, based on Matrix Mechanism [28], for answering workloads of predicate counting queries.

The notion of local differential privacy (LDP) was introduced in [26]. Early works on LDP mainly focus on estimating frequencies of values of an attribute having a categorical domain [1, 5, 16, 45, 54]. Wang et al. [45] investigate these approaches and conclude that OLH is the state-of-the-art for a relatively large domain. More recently, for this problem, Wang et al. [44] propose a novel wheel mechanism, which has a same variance as OLH. For ordinal or numerical attributes, studies are mostly concentrated on mean estimation [12, 13, 42]. Only several works investigate range queries. For answering 1-D range queries on a singe attribute, Cormode et al. [9] extend the ideas of hierarchical intervals and Haar wavelet transform to the LDP setting. Li et al. [29] propose the Square Wave (SW) approach for reconstructing the distribution of an ordinal attribute. We have extended SW to answer multi-dimensional range queries in Section 3.5 and examined its performance. The most closely related work for answering multi-dimensional range queries is HIO proposed by Wang et al. [46], which is designed for multi-dimensional analytical queries. We have considered HIO as a baseline approach and proposed an improvement of it in Section 3.

In addition, approaches [8, 37, 56] for marginal release under LDP can be also used to answer multi-dimensional queries. Ren et al. [37] generalizes the Expectation Maximization algorithm for estimating joint distribution of two attributes. Cormode et al. [8] refine and analyze how to release marginals via transformations under LDP. CALM proposed by Zhang et al. [56] is the state-of-art for marginal release under LDP. It adapts the ideas of consistency enforcement and maximum entropy estimation from PriView [34] to LDP setting. We have also analysed its performance in handling our problem in Section 3.2.

LDP has been also applied to support other data analysis tasks, such as collecting frequent items or itemsets [4, 6, 19, 23, 35, 41, 43, 47], locations [7, 17], key-value data [18, 55], social graphs [36, 39], linear query answers [3, 15, 30], telemetry data [11], preference rankings [51] and evolving data [24]. However, since they work on the problems that are different from ours, their approaches are not suitable for answering multi-dimensional range queries.

7 CONCLUSIONS

In this paper, we present TDG and HDG, two novel approaches for answering multi-dimensional range queries under LDP. We claim that TDG and HDG satisfy ε-LDP. We theoretically analyse different sources of errors and provide a guideline for properly choosing granularities. Our results demonstrate the effectiveness of HDG.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 61872045 and No. 62072052), the Innovation Research Group Project of NSFC (61921003), and the National Science Foundation (Grant No. 1640374 and No. 1931443). Professors Ninghui Li and Sen Su are the co-corresponding authors.

REFERENCES

[1] Jayadev Acharya, Ziteng Sun, and Huanyu Zhang. 2019. Hadamard Response: Estimating Distributions Privately, Efficiently, and with Little Communication. In *AISTATS*, Vol. 89. PMLR, 1120–1129.

[2] Sanjeev Arora, Elad Hazan, and Satyen Kale. 2012. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory Comput.* 8, 1 (2012), 121–164.

[3] Raef Bassily. 2019. Linear Queries Estimation with Local Differential Privacy. In *AISTATS (Proceedings of Machine Learning Research)*, Vol. 89. PMLR, 721–729.

[4] Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Guha Thakurta. 2017. Practical Locally Private Heavy Hitters. In *NIPS*. 2288–2296.

[5] Raef Bassily and Adam D. Smith. 2015. Local, Private, Efficient Protocols for Succinct Histograms. In *STOC*. ACM, 127–135.

[6] Mark Bun, Jelani Nelson, and Uri Stemmer. 2018. Heavy Hitters and the Structure of Local Privacy. In *PODS*. ACM, 435–447.

[7] Rui Chen, Haoran Li, A. Kai Qin, Shiva Prasad Kasiviswanathan, and Hongxia Jin. 2016. Private spatial data aggregation in the local setting. In *ICDE*. IEEE Computer Society, 289–300.

[8] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2018. Marginal Release Under Local Differential Privacy. In *SIGMOD*. ACM, 131–146.

[9] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2019. Answering Range Queries Under Local Differential Privacy. *PVLDB* 12, 10 (2019), 1126–1138.

[10] Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. 2012. Differentially Private Spatial Decompositions. In *ICDE*. IEEE Computer Society, 20–31.

[11] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting Telemetry Data Privately. In *NIPS*. 3571–3580.

[12] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local Privacy and Statistical Minimax Rates. In *FOCS*. IEEE Computer Society, 429–438.

[13] John C. Duchi, Martin J. Wainwright, and Michael I. Jordan. 2013. Local Privacy and Minimax Bounds: Sharp Rates for Probability Estimation. In *NIPS*. 1529–1537.

[14] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *TCC (Lecture Notes in Computer Science)*, Vol. 3876. Springer, 265–284.

[15] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. 2020. The power of factorization mechanisms in local and central differential privacy. In *STOC*. ACM, 425–438.

[16] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *CCS*. ACM, 1054–1067.

[17] Xiaolan Gu, Ming Li, Yang Cao, and Li Xiong. 2019. Supporting Both Range Queries and Frequency Estimation with Local Differential Privacy. In *CNS*. IEEE, 124–132.

[18] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. 2020. PCKV: Locally Differentially Private Correlated Key-Value Data Collection with Optimized Utility. In *USENIX Security*. USENIX Association, 967–984.

[19] Xiaolan Gu, Ming Li, Li Xiong, and Yang Cao. 2020. Providing Input-Discriminative Protection for Local Differential Privacy. In *ICDE*. IEEE, 505–516.

[20] Moritz Hardt, Katrina Ligett, and Frank McSherry. 2012. A Simple and Practical Algorithm for Differentially Private Data Release. In *NIPS*. 2348–2356.

[21] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, and Dan Zhang. 2016. Principled Evaluation of Differentially Private Algorithms using DPBench. In *SIGMOD*. ACM, 139–154.

[22] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. 2010. Boosting the Accuracy of Differentially Private Histograms Through Consistency. *PVLDB* 3, 1 (2010), 1021–1032.

[23] Justin Hsu, Sanjeev Khanna, and Aaron Roth. 2012. Distributed Private Heavy Hitters. In *ICALP*, Vol. 7391. Springer, 461–472.

[24] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. 2018. Local Differential Privacy for Evolving Data. In *NIPS*. 2381–2390.

[25] Kaggle. [n.d.]. Big Five Personality Test. http://www.kaggle.com/tunguz/big-five-personality-test/data.

[26] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. 2008. What Can We Learn Privately?. In *FOCS*. IEEE Computer Society, 531–540.

[27] Chao Li, Michael Hay, Gerome Miklau, and Yue Wang. 2014. A Data- and Workload-Aware Query Answering Algorithm for Range Queries Under Differential Privacy. *PVLDB* 7, 5 (2014), 341–352.

[28] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. 2010. Optimizing linear counting queries under differential privacy. In *PODS*. ACM, 123–134.

[29] Zitao Li, Tianhao Wang, Milan Lopuhaä-Zwakenberg, Ninghui Li, and Boris Skoric. 2020. Estimating Numerical Distributions under Local Differential Privacy. In *SIGMOD*. ACM, 621–635.

[30] Ryan McKenna, Raj Kumar Maity, Arya Mazumdar, and Gerome Miklau. 2020. A workload-adaptive mechanism for linear queries under local differential privacy. *PVLDB* 13, 11 (2020), 1905–1918.

[31] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. 2018. Optimizing error of high-dimensional statistical queries under differential privacy. *PVLDB* 11, 10 (2018), 1206–1219.

[32] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2013. Differentially private grids for geospatial data. In *ICDE*. IEEE Computer Society, 757–768.

[33] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2013. Understanding Hierarchical Methods for Differentially Private Histograms. *PVLDB* 6, 14 (2013), 1954–1965.

[34] Wahbeh H. Qardaji, Weining Yang, and Ninghui Li. 2014. PriView: practical differentially private release of marginal contingency tables. In *SIGMOD*. ACM, 1435–1446.

[35] Zhan Qin, Yin Yang, Ting Yu, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2016. Heavy Hitter Estimation over Set-Valued Data with Local Differential Privacy. In *CCS*. ACM, 192–203.

[36] Zhan Qin, Ting Yu, Yin Yang, Issa Khalil, Xiaokui Xiao, and Kui Ren. 2017. Generating Synthetic Decentralized Social Graphs with Local Differential Privacy. In *CCS*. ACM, 425–438.

[37] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and Philip S. Yu. 2018. LoPub: High-Dimensional Crowdsourced Data Publication With Local Differential Privacy. *TIFS* 13, 9 (2018), 2151–2166.

[38] Steven Ruggles, J. Trent Alexander, Katie Genadek, Ronald Goeken, Matthew B. Schroeder, and Matthew Sobek. 2010. Integrated Public Use Microdata Series: Version 5.0 [Machine-readable database].

[39] Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Wendy Hui Wang, and Ting Yu. 2019. Analyzing Subgraph Statistics from Extended Local Views with Decentralized Differential Privacy. In *CCS*. ACM, 703–717.

[40] Apple Differential Privacy Team. 2017. Learning with Privacy at Scale, available at http://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf.

[41] Ning Wang, Xiaokui Xiao, Yin Yang, Ta Duy Hoang, Hyejin Shin, Junbum Shin, and Ge Yu. 2018. PrivTrie: Effective Frequent Term Discovery under Local Differential Privacy. In *ICDE*. IEEE Computer Society, 821–832.

[42] Ning Wang, Xiaokui Xiao, Yin Yang, Jun Zhao, Siu Cheung Hui, Hyejin Shin, Junbum Shin, and Ge Yu. 2019. Collecting and Analyzing Multidimensional Data with Local Differential Privacy. In *ICDE*. IEEE, 638–649.

[43] Shaowei Wang, Liusheng Huang, Yiwen Nie, Pengzhan Wang, Hongli Xu, and Wei Yang. 2018. PrivSet: Set-Valued Data Analyses with Locale Differential Privacy. In *INFOCOM*. IEEE, 1088–1096.

[44] Shaowei Wang, Yuqiu Qian, Jiachun Du, Wei Yang, Liusheng Huang, and Hongli Xu. 2020. Set-valued Data Publication with Local Privacy: Tight Error Bounds and Efficient Mechanisms. *PVLDB* 13, 8 (2020), 1234–1247.

[45] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *USENIX Security*. USENIX Association, 729–745.

[46] Tianhao Wang, Bolin Ding, Jingren Zhou, Cheng Hong, Zhicong Huang, Ninghui Li, and Somesh Jha. 2019. Answering Multi-Dimensional Analytical Queries under Local Differential Privacy. In *SIGMOD*. ACM, 159–176.

[47] Tianhao Wang, Ninghui Li, and Somesh Jha. 2018. Locally Differentially Private Frequent Itemset Mining. In *SP*. IEEE Computer Society, 127–143.

[48] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. 2020. Locally Differentially Private Frequency Estimation with Consistency. In *NDSS*. The Internet Society.

[49] Stanley L. Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

[50] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. 2010. Differential privacy via wavelet transforms. In *ICDE*. IEEE Computer Society, 225–236.

[51] Jianyu Yang, Xiang Cheng, Sen Su, Rui Chen, Qiyu Ren, and Yuhan Liu. 2019. Collecting Preference Rankings Under Local Differential Privacy. In *ICDE*. IEEE, 1598–1601.

[52] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Answering Multi-Dimensional Range Queries under Local Differential Privacy. *CoRR* abs/2009.06538 (2020). https://arxiv.org/abs/2009.06538

[53] Jianyu Yang, Tianhao Wang, Ninghui Li, Xiang Cheng, and Sen Su. 2020. Source Code of Approaches. [Online]. http://github.com/YangJianyu-bupt/privmdr.

[54] Min Ye and Alexander Barg. 2018. Optimal Schemes for Discrete Distribution Estimation Under Locally Differential Privacy. *IEEE Trans. Inf. Theory* 64, 8 (2018), 5662–5676.

[55] Qingqing Ye, Haibo Hu, Xiaofeng Meng, and Huadi Zheng. 2019. PrivKV: Key-Value Data Collection with Local Differential Privacy. In *SP*. IEEE, 317–331.

[56] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *CCS*. ACM, 212–229.